Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Омский государственный технический университет»

Факультет Информационных технологий и компьютерных систем

Кафедра Информатика и вычислительная техника

РАСЧЕТНО-ГРАФИЧЕСКАЯ РАБОТА

«Чат»

По дисциплине «Проектирование и тестирование программного обеспечения»

Выполнил:
студент группы ПИ-171
Селькина Л.М.
5.06.19

Принял:
ассистент Фахрутдинов А.Р.
5.06.19

Омск, 2019

СОДЕРЖАНИЕ

Введение

Слово Чат (*chat*) по-английски означает разговор. В сети Интернет Чат - это такая страница, где можно в реальном времени общаться с другими посетителями. Каждый присутствующий, при входе вписывает свой ник (от англ. *nickname*), т.е. прозвище, которым он хочет, чтобы его называли. Дальше смотрите на экран, который будет периодически стирать все, что на нем написано и печатать те фразы, которые за последние несколько секунд послали другие посетители. Каждый может, как просто наблюдать, так и сам писать сообщения другим.

Задача данной программы сымитировать локальный чат, работающий без интернета. Она является локальным чатом, которым может одновременно пользоваться множество клиентов. Так же предусмотрен чат-бот, отвечающий на имя «Крио» перед сообщением, адресованных ему.

Данное приложение разрабатывалось на языке C# при помощи Windows Form.

После ввода имени текстовые поля и кнопка становятся не активными.

1 Формулировка задания для РГР

Требовалось разработать чат, который должен:

1. Иметь отдельное графическое окно;
2. Иметь поле для ввода имени;
3. Иметь кнопку для подтверждения выбора имени;
4. Иметь поле для истории чата;
5. Иметь поле для написания сообщения;
6. Иметь кнопку для подтверждения отправки сообщения на сервер;
7. Отправлять сообщение всем пользователем через сервер (сокет-соединение);
8. Предоставлять возможность общаться и обучать простейшего чат-бота.

Тестирование данного приложения должно:

1. Проверять правильность вывода в одной из подпрограмм;
2. Проверять правильность подсчета определенных значений на стороне сервера.

2 Разработка РГР

2.1 Детали разработки

Для реализации программы был выбран язык C#, интерфейс был выполнен с помощью Windows Form. Для программирования самого приложения было отдано предпочтение среде разработки ПО – Microsoft Visual Studio 2019.

2.3 Разработка интерфейса

В данной работе использовался код на языке программирования C#. Использовалось «Приложение Windows Forms», чтобы отобразить окно с элементами чата (рисунок 1).

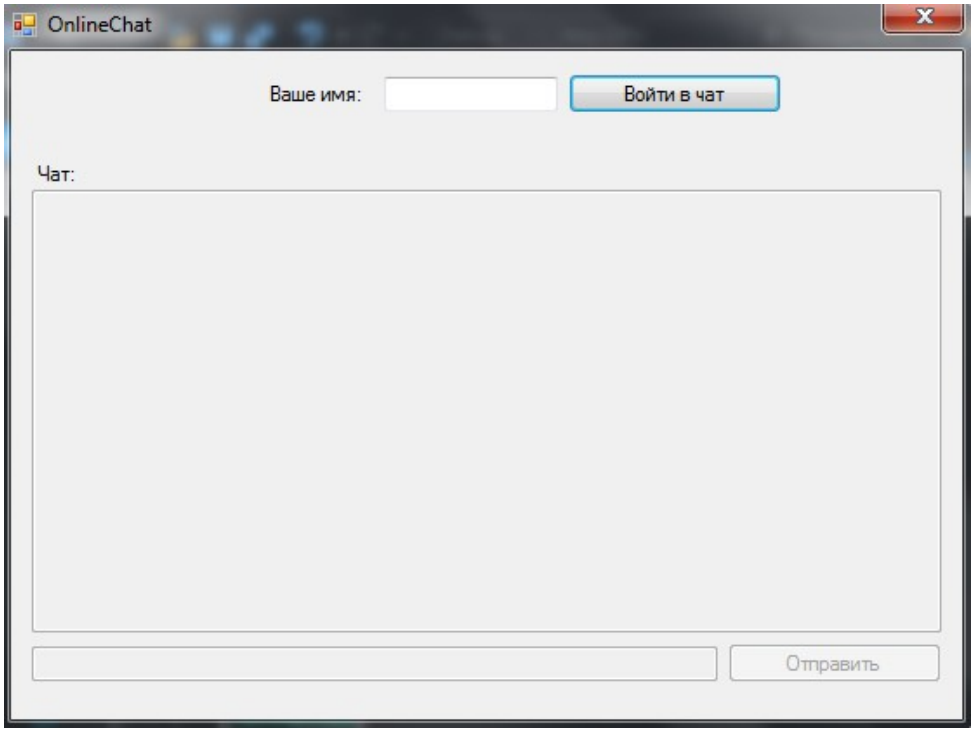

Рисунок 1 – Главное окно

Все действия, такие как ввод имени, подтверждение выбора никнейма, написание и отправку сообщений на сервер делаются при помощи элементов графического окна.

2.3 Ход работы

Программа состоит из двух видов частей: сервер и клиент. В ней реализована многопоточность, поэтому подключаться к серверу может несколько клиентов.

Запускать программу нужно с сервера. Некоторые действия, например, запуск сервера, регистрация новых подключений и отчет об отправке сообщений одним из клиентов, отслеживаются и выводятся на консоль.

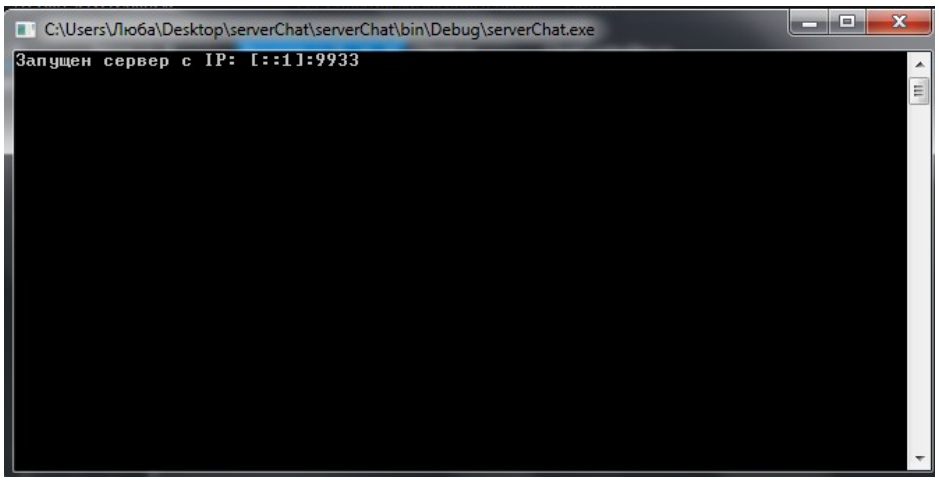

Рисунок 2 – Консоль сервера

Далее запускается клиент. При его подключении в сервере появляется оповещение об этом.

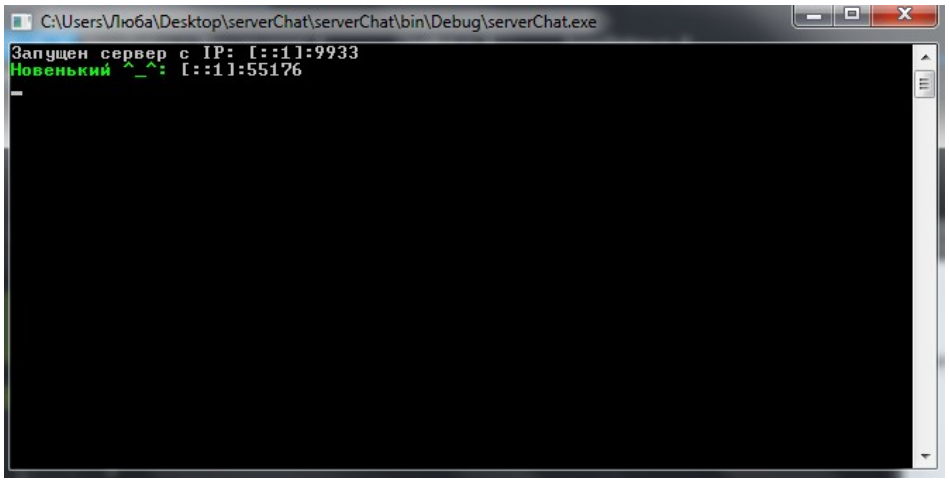

Рисунок 3 – Оповещение о подключении на сервере

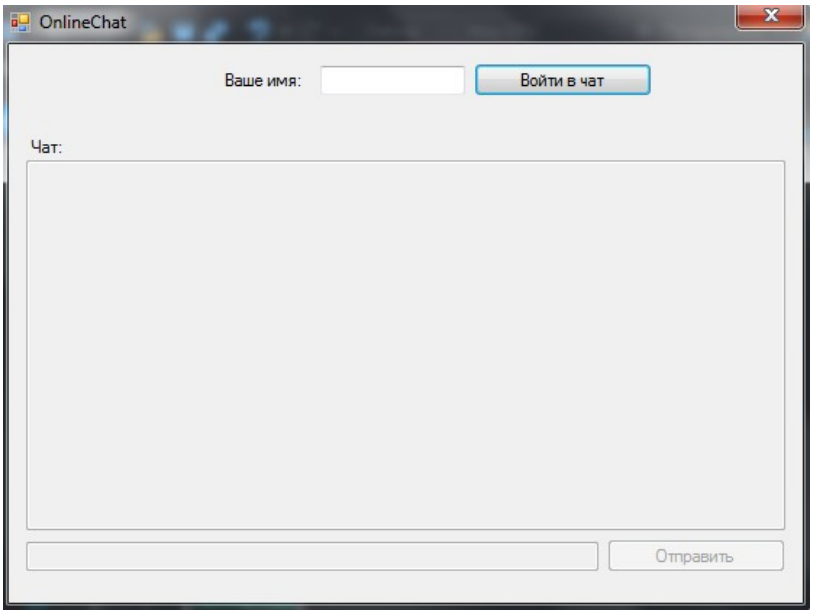

Рисунок 4 – Графическое меню у клиента

Далее вводится имя пользователя и отправляется сообщение.

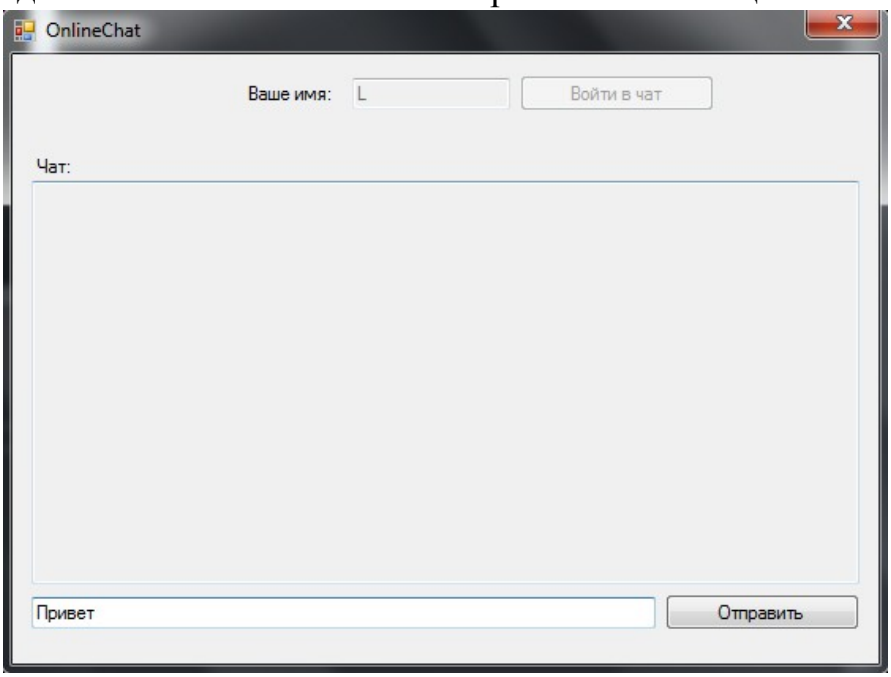

Рисунок 5 – Ввод и подтверждение имени

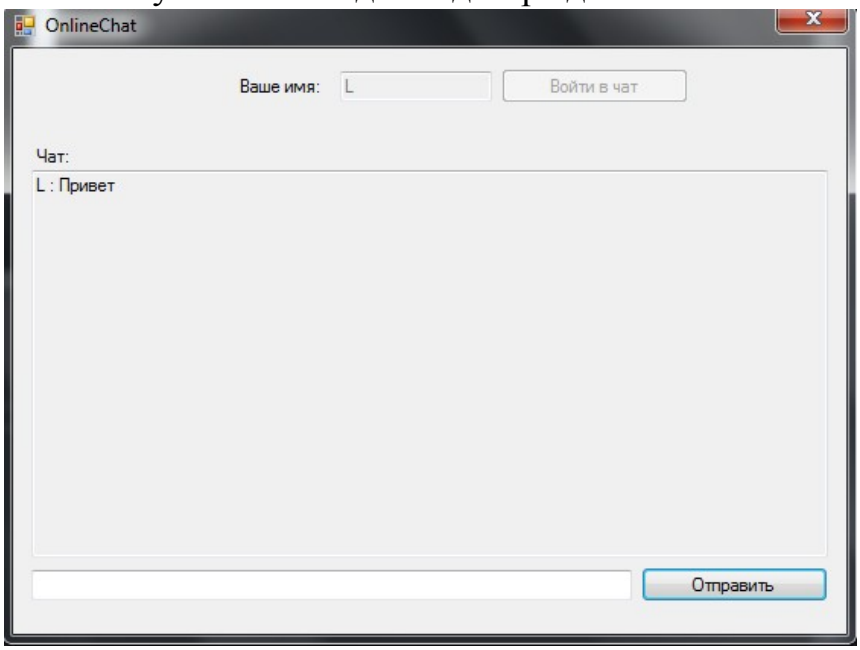

Рисунок 6 – Ввод и отправка сообщения в чате

Отправка сообщения фиксируется на сервере.

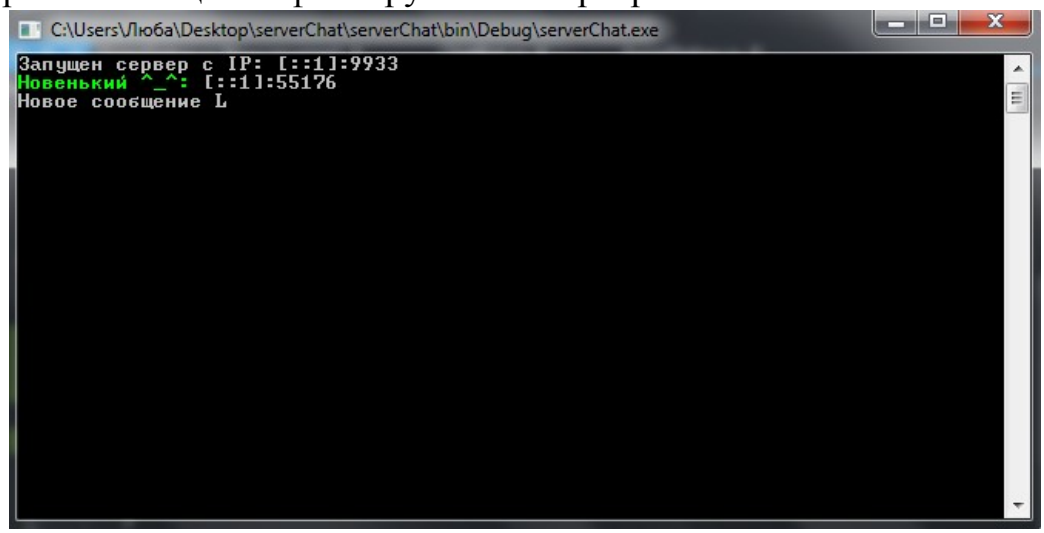

Рисунок 7 – Регистрация входа на сервере

Можно пообщаться с ботом и обучить его.

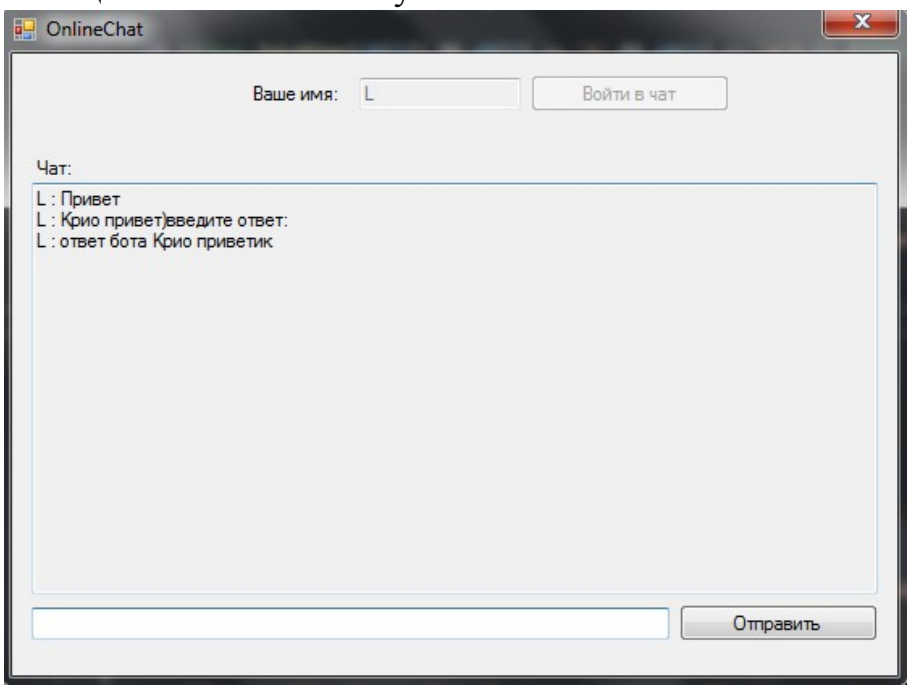

Рисунок 8 – Первое обращение к боту

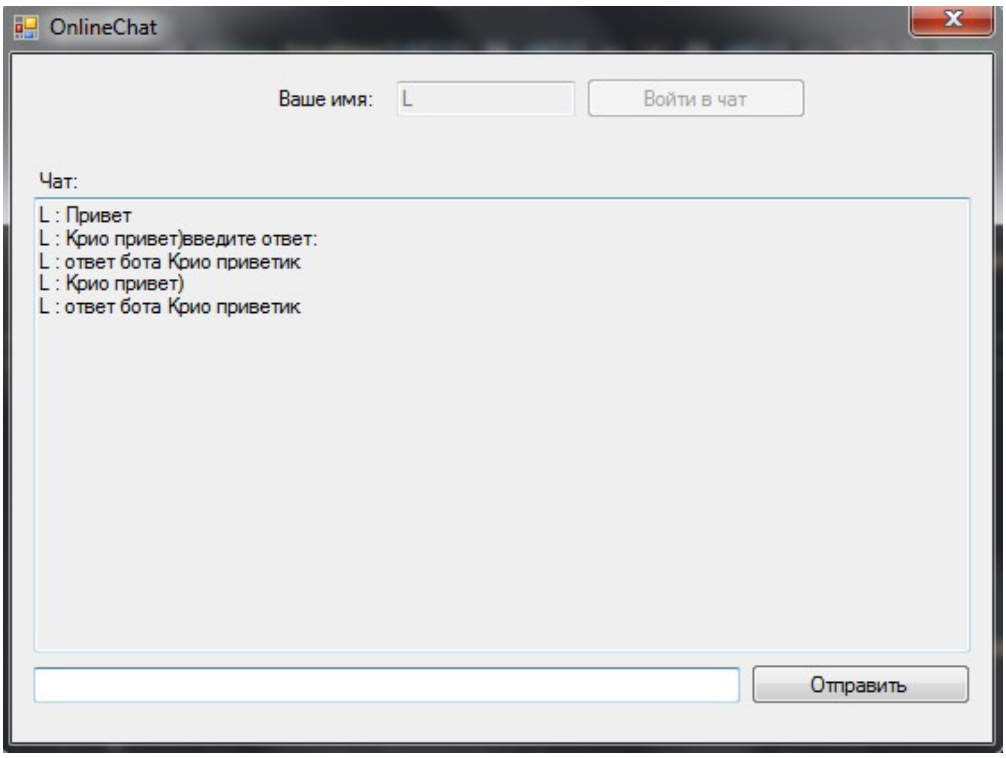

Рисунок 9 – Второе обращение к боту

Подключается второй клиент.

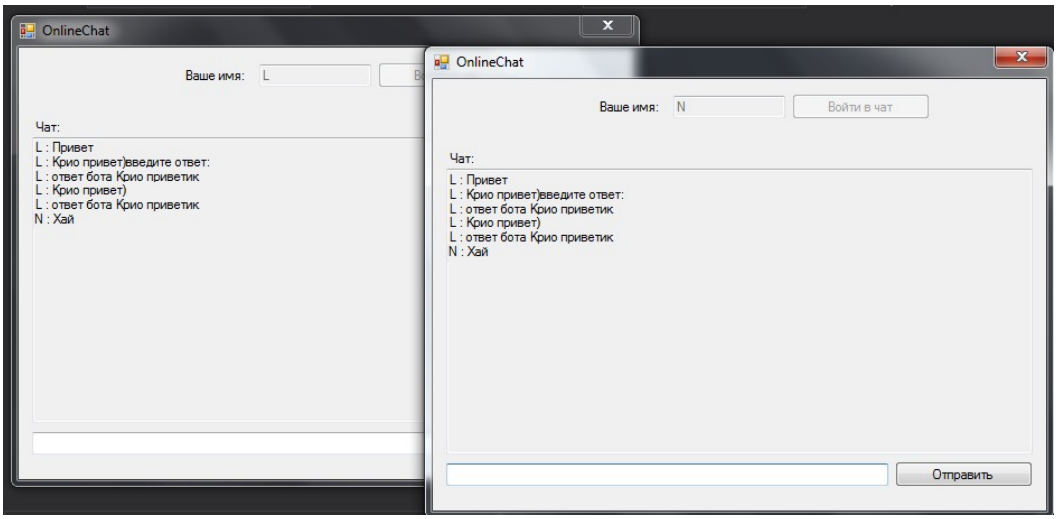

Рисунок 10 – Второй пользователь

3 Тестирование

3.1 Тест-кейс

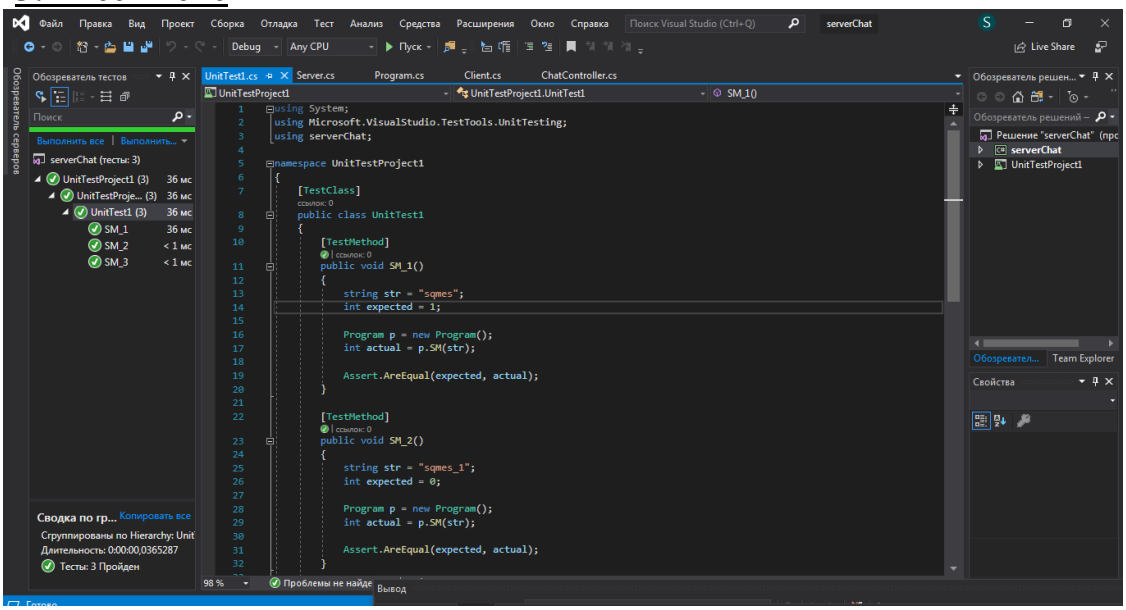

Рисунок 5 – Тест 1-3

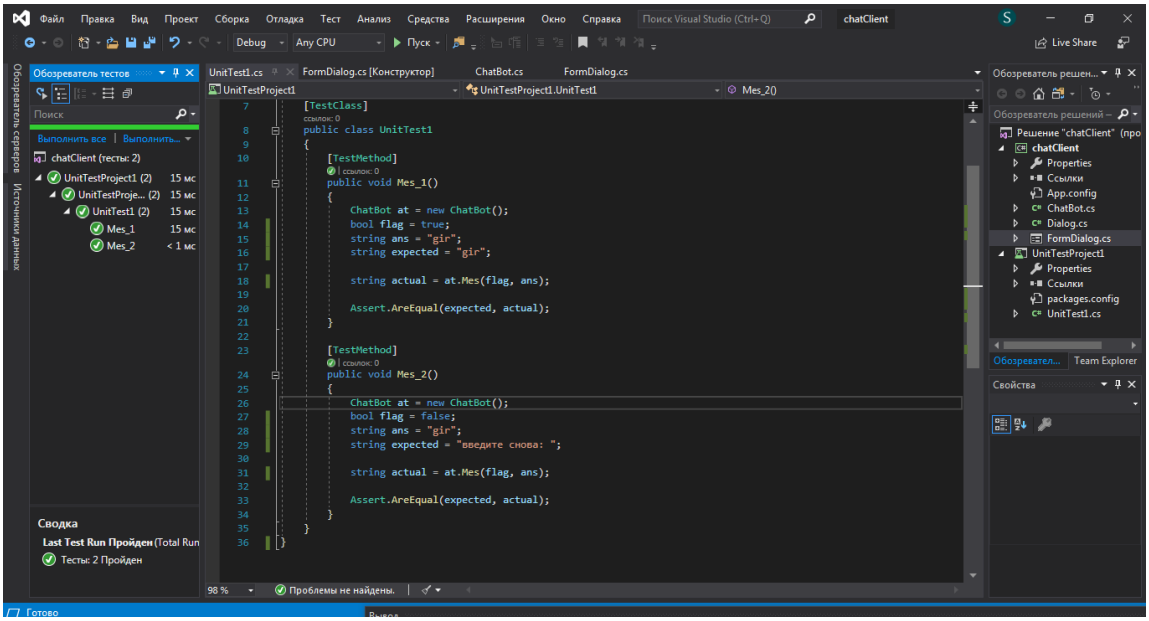

Рисунок 6 – Тест 4-5

Заключение

В результате выполнения проекта был разработан проект, реализующий функции локального чата.

Плюсы:

- удобность за счет Windows form.
- не ограниченность по времени использования;
- не ограниченность в плане занимаемой памяти;
- общение не только с участниками чата, но и с чат-ботом;
- возможность обучения чат-бота.

Минусы:

- Чат-бот зависит от файла, в котором хранятся его данные;
- Невозможно повторно изменить имя.

Библиографический список

1. Жилин, С. А. Информатика. Теория и практика решения задач. Курс углубленного изучения / С. А. Жилин, И. Б. Жилина. – М. : РКНК, 2001. – 301 с.

2. Златопольский, Д. М. Сборник задач по программированию / Д. М. Златопольский. – 2-е изд., перераб. и доп. — СПб. : БХВПетербург, 2007. — 240 с.

3. Материалы библиотеки msdn. URL www.msdn.microsoft.com. Дата актуальности: 12.03.2013.

Приложение А

```
//Пакет ServerChat
//Server
using System;
using System.Collections.Generic;
using System.Net.Sockets;
using System.Text;

namespace serverChat
{
    public static class Server
    {
        public static List<Client> Clients = new List<Client>();
        public static void NewClient(Socket handle) // подключение нового клиента
        {
            try
            {
                Client newClient = new Client(handle);
                Clients.Add(newClient);
                Console.ForegroundColor = ConsoleColor.Green;
                Console.Write("Новенький ^_^:");
                Console.ResetColor();
                Console.WriteLine(" {0}", handle.RemoteEndPoint);
            }
            catch (Exception e)
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Ошибка соединения с новым клиентом: {0}", e.Message);
                Console.ResetColor();
            }
        }


        public static void EndClient(Client client) // отключение клиента
        {
            try
            {
                client.End();
                Clients.Remove(client);
                Console.ForegroundColor = ConsoleColor.Yellow;
                Console.WriteLine("Пользователь {0} вышел из чата", client.UserName);
                Console.ResetColor();
            }
            catch (Exception e)
            {
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Ошибка при выходе пользователя: {0}", e.Message);
                Console.ResetColor();
            }
        }


        public static void UpdateAllChats() //обнова всех окон чатов при new message
        {
            try
            {
                int count_users = Clients.Count; //сколько в листе клиентов
                for (int i = 0; i < count_users; i++)
                {
                    Clients[i].UpdateChat(); //из класса Client
                }
            }
            catch (Exception e)
            {
```

```
                Console.ForegroundColor = ConsoleColor.Red;
                Console.WriteLine("Ошибка обновления чатов: {0}", e.Message);
                Console.ResetColor();
            }
        }

    }
}
```

//Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace serverChat
{
    public class Program
    {
        private const string localhost = "localhost";
        private const int serverPort = 9933;
        private static Thread serverThread; //сервашные потоки


        public static int Main()
        {
            string str = "sqmes";
            Program p = new Program();

            serverThread = new Thread(startServer);
            serverThread.IsBackground = true;
            serverThread.Start(); //поток сервака
            int lp = p.SM(str);

            while (true)
            {
                handlerCommands(Console.ReadLine()); // ждем сообщения
                str = "sqmes_1";
            }
        }

        public int SM(string str)
        {
            int i = 0;
            if (str == "sqmes")
            {
                i++;
            }
            return i;
        }

        public static void handlerCommands(string cmd) //обработка команды
        {
            cmd = cmd.ToLower();
            if (cmd.Contains("/getusers"))
            {
                int count_users = Server.Clients.Count();
                for (int i = 0; i < count_users; i++)
                {
```

```
                    Console.WriteLine("{0}: {1}", i, Server.Clients[i].UserName);
                }
            }
        }


        private static void startServer() // ачало работы сервака; адресовка;
        {
            IPHostEntry ip_host = Dns.GetHostEntry(localhost);
            IPAddress ip_address = ip_host.AddressList[0];
            IPEndPoint ipEndPoint = new IPEndPoint(ip_address, serverPort);
            Socket socket = new Socket(ip_address.AddressFamily, SocketType.Stream,
ProtocolType.Tcp);
            socket.Bind(ipEndPoint);
            socket.Listen(1000);
            Console.WriteLine("Запущен сервер с IP: {0}", ipEndPoint);
            while(true) // list пользователей
            {
                try
                {
                    Socket user = socket.Accept();
                    Server.NewClient(user);
                }
                catch (Exception e)
                {
                    Console.WriteLine("Ошибка создания клиента: {0}", e.Message);
                }
            }

        }
    }
}
```

//Client

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Sockets;
using System.Threading;

namespace serverChat
{
    public class Client
    {
        private string userName;
        private Socket handler;
        private Thread userThread;
        public Client(Socket socket) //стартует поток клиента и заказываем обработочку
        {
            handler = socket;
            userThread = new Thread(listner);
            userThread.IsBackground = true;
            userThread.Start();
        }

        public string UserName // чисто для передачи Server и Program
        {
            get { return userName; }
        }

        private void listner()
        {
```

```
            while (true)
            {
                try
                {
                    byte[] buffer = new byte[1024];
                    int bytesRec = handler.Receive(buffer); //получает в буффер данные из
сокета
                    string data = Encoding.UTF8.GetString(buffer, 0, bytesRec); // чтобы
читабельно было
                    handleCommand(data); //для обработочки
                }
                catch
                {
                    Server.EndClient(this);
                    return;
                } //молча перекрыть клиента
            }
        }


        public void End()
        {
            try
            {
                handler.Close(); //закрыть сокет
                userThread.Abort(); //закрытие потока
            }
            catch (Exception e)
            {
                Console.WriteLine("Ошибка закрытия: {0}", e.Message);
            }
        }



        private void handleCommand(string data)
        {
            if (data.Contains("#setname"))
            {
                userName = data.Split('&')[1];
                UpdateChat();
                return;
            }
            if (data.Contains("#newmsg"))
            {
                string message = data.Split('&')[1];
                ChatController.AddMessage(userName,message);
                return;
            }
        }
        public void UpdateChat() //обновить чат отправителя
        {
            string command = ChatController.GetChat();

            try
            {
                int bytesSent = handler.Send(Encoding.UTF8.GetBytes(command));
            }
            catch (Exception e)
            {
                Console.WriteLine("Ошибка команды: {0}", e.Message);
Server.EndClient(this);
            }
        }
    }
```

```
}
```

//ChatController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace serverChat
{
    public static class ChatController
    {
        private const int maxMessage = 1000;
        public static List<message> Chat = new List<message>();
        public struct message
        {
            public string userName;
            public string data;
            public message(string name, string msg)
            {
                userName = name;
                data = msg;
            }
        }
        public static void AddMessage(string userName, string msg)
        {
            try
            {
                if (string.IsNullOrEmpty(userName) || string.IsNullOrEmpty(msg)) return;
                int countMessages = Chat.Count;
                if (countMessages > maxMessage) ClearChat();
                message newMessage = new message(userName, msg);
                Chat.Add(newMessage);
                Console.WriteLine("Новое сообщение {0}", userName);
                Server.UpdateAllChats();
            }
            catch (Exception e) { Console.WriteLine("Ошибка добавлением сообщения: {0}",
e.Message); }
        }


        public static void ClearChat()
        {
            Chat.Clear();
        }


        public static string GetChat()
        {
            try
            {
                string data = "#updatechat&";
                int countMessages = Chat.Count;
                if (countMessages <= 0) return string.Empty;
                for (int i = 0; i < countMessages; i++)
                {
                    data += String.Format("{0}~{1}|", Chat[i].userName,
Chat[i].data);//преобразование строки и вставка в другую
                }
                return data;
            }
            catch (Exception e)
```

```
            {
                Console.WriteLine("Ошибка в getChat: {0}", e.Message);
                return string.Empty;
            }
        }
    }
}
```

```
//Пакет chatClient
//ChatBot
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;

namespace chatClient
{
    public class ChatBot
    {
        string q, path; //вопрос; путь; ответпользователя (обучение)
        List<string> sempls = new List<string>();//вопрос-ответ

        //генерация ответа
        public int Que(string qw, string pat)
        {
            int h = 1;
            path = pat;

            try
            {
                sempls.AddRange(File.ReadAllLines(path));
            }
            catch
            {

            }

            q = qw;
            for (int i = 1; i < sempls.Count; i += 2)
            {
                if (qw == sempls[i])
                {
                    h = 0;
                }
            }

            if (h == 1)
            {
                sempls.Add(qw);
                File.WriteAllLines(path, sempls.ToArray());
            }
            return h;
        }


        public string Ans(string ans, string pat)
        {
            path = pat;
            try
            {
```

```
                sempls.AddRange(File.ReadAllLines(path));
            }
            catch
            {

            }

            path = pat;
            sempls.Add(ans);
            File.WriteAllLines(path, sempls.ToArray());
            return ans;
        }

        public string Anss(string ans, string pat)
        {
            ChatBot at = new ChatBot();
            path = pat;

            try
            {
                sempls.AddRange(File.ReadAllLines(path));
            }
            catch
            {

            }

            ans = ans.ToLower();
            for (int i = 1; i < sempls.Count; i += 2)
            {
                if (ans == sempls[i])
                {
                    return at.Mes(true, ans);
                }
            }

            return at.Mes(false, "");
        }

        public string Mes(bool flag, string ans)
        {
            string str;
            if (flag)
            {
                str = ans;
            }
            else
            {
                str = "введите снова: ";
            }
            return str;
        }

    }
}
```

```
//FormDialog
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Net.Sockets;
using System.Net;
using System.Threading;

namespace chatClient
{
    public partial class chatForm : Form
    {
        private delegate void printer(string data);
        private delegate void cleaner();
        printer Printer;
        cleaner Cleaner;
        private Socket serverSocket;
        private Thread clientThread;
        private const string serverHost = "localhost";
        private const int serverPort = 9933;

        public chatForm()
        {
            InitializeComponent();
            Printer = new printer(print);
            Cleaner = new cleaner(clearChat);
            connect();
            clientThread = new Thread(listner);
            clientThread.IsBackground = true;
            clientThread.Start();
        }


        private void listner()
        {
            while (serverSocket.Connected)
            {
                byte[] buffer = new byte[8196];
                int bytesRec = serverSocket.Receive(buffer);
                string data = Encoding.UTF8.GetString(buffer, 0, bytesRec);
                if (data.Contains("#updatechat"))
                {
                    UpdateChat(data); //обнова чата
                    continue;
                }
            }
        }
        private void connect()
        {
            try
            {
                IPHostEntry ipHost = Dns.GetHostEntry(serverHost);
                IPAddress ipAddress = ipHost.AddressList[0];
                IPEndPoint ipEndPoint = new IPEndPoint(ipAddress, serverPort);
                serverSocket = new Socket(ipAddress.AddressFamily, SocketType.Stream,
ProtocolType.Tcp);
                serverSocket.Connect(ipEndPoint);
            }
            catch { print("Сервер недоступен!"); }
```

```
        }

        private void clearChat()
        {
            if (this.InvokeRequired)
            {
                this.Invoke(Cleaner);
                return;
            }
            chatBox.Clear();
        }
        private void UpdateChat(string data)
        {
            //#updatechat&userName~data|username~data
            clearChat();
            string[] messages = data.Split('&')[1].Split('|');
            int countMessages = messages.Length;
            if (countMessages <= 0) return;
            for (int i = 0; i < countMessages; i++)
            {
                try
                {
                    if (string.IsNullOrEmpty(messages[i])) continue;
                    print(String.Format("{0} : {1}", messages[i].Split('~')[0],
messages[i].Split('~')[1]));
                }
                catch { continue; }
            }
        }
        private void send(string data) //отправка на сервер
        {
            try
            {
                byte[] buffer = Encoding.UTF8.GetBytes(data);
                int bytesSent = serverSocket.Send(buffer);
            }
            catch
            {
                print("Связь с сервером прервалась...");
            }
        }
        private void print(string msg)
        {

            if (this.InvokeRequired)
            {
                this.Invoke(Printer, msg);
                return;
            }
            if (chatBox.Text.Length == 0)
                chatBox.AppendText(msg);
            else
                chatBox.AppendText(Environment.NewLine + msg);
        }

        private void enterChat_Click(object sender, EventArgs e)
        {
            string Name = userName.Text;
            if (string.IsNullOrEmpty(Name)) return;
            send("#setname&" + Name); // отправка имени на сервер
            chatBox.Enabled = true;
            chat_msg.Enabled = true;
            chat_send.Enabled = true;
            userName.Enabled = false;
            enterChat.Enabled = false;
```

```
        }

        private void chat_send_Click(object sender, EventArgs e)
        {
            sendMessage();
        }

        public static bool flag = false;
        private void sendMessage()
        {
            ChatBot bot = new ChatBot();
            try
            {
                int c = 0;
                string data = chat_msg.Text;
                if (string.IsNullOrEmpty(data)) return;
                if (data.Contains("Крио") || data.Contains("крио"))//вопрос точно должен
содержать один знак или цифру(главное чтобы все не были буквамии пробелами!!!!)
                {
                    char[] ch = data.ToCharArray();
                    for (int i = 0; i < ch.Length; i++)
                    {
                        if ((Char.IsLetter(ch[i]) && ch[i] >= 'А' && ch[i] <= 'я') ||
(Char.IsLetter(ch[i]) && ((ch[i] == 'Ё') || (ch[i] == 'ё'))) || ch[i] == ' ')
                        {
                            c++;
                        }
                    }
                    if (c == ch.Length - 1)
                    {
                        if (bot.Que(data,
@"C:\Users\Люба\Desktop\chatClient\question_answer.txt") == 1)
                        {
                            flag = true;//запись ответа сразу без проверок
                            send("#newmsg&" + data + "введите ответ: ");
                        }
                        else
                        {
                            flag = false;
                            send("#newmsg&" + data);
                        }
                    }
                    if (c != ch.Length - 1 && flag == true) //ответа в базе нет
                    {
                        send("#newmsg&" + "ответ бота " + bot.Ans(data,
@"C:\Users\Люба\Desktop\chatClient\question_answer.txt"));
                    }
                    if (c != ch.Length - 1 && flag == false) //ответ есть
                    {
                        send("#newmsg&" + "ответ бота " + bot.Anss(data,
@"C:\Users\Люба\Desktop\chatClient\question_answer.txt"));
                    }
                }
                else
                {
                    send("#newmsg&" + data);//отправка сообщения на сервер
                }
                chat_msg.Text = string.Empty;
            }
            catch { MessageBox.Show("Ошибка при отправке сообщения!"); }
        }

        private void chatBox_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyData == Keys.Enter)
```

```
            sendMessage();
        }

        private void chat_msg_KeyDown(object sender, KeyEventArgs e)
        {
            if (e.KeyData == Keys.Enter)
                sendMessage();
        }

        private void Gui_userName_Click(object sender, EventArgs e)
        {

        }

        private void VScrollBar1_Scroll(object sender, ScrollEventArgs e1)
        {
        }

        private void ChatBox_TextChanged(object sender, EventArgs e)
        {

        }

        private void Chat_msg_TextChanged(object sender, EventArgs e)
        {

        }

        private void UserName_TextChanged(object sender, EventArgs e)
        {

        }

        private void Gui_chat_Click(object sender, EventArgs e)
        {

        }
    }
}
```

Приложение Б

//Тест 1-3(пакет serverChat)

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace serverChat
{
    public class Program
    {
        private const string localhost = "localhost";
        private const int serverPort = 9933;
        private static Thread serverThread; //сервашные потоки


        public static int Main()
        {
            string str = "sqmes";
            Program p = new Program();

            serverThread = new Thread(startServer);
            serverThread.IsBackground = true;
            serverThread.Start(); //поток сервака
            int lp = p.SM(str);

            while (true)
            {
                handlerCommands(Console.ReadLine()); // ждем сообщения
                str = "sqmes_1";
            }
        }

        public int SM(string str)
        {
            int i = 0;
            if (str == "sqmes")
            {
                i++;
            }
            return i;
        }

        public static void handlerCommands(string cmd) //обработка команды
        {
            cmd = cmd.ToLower();
            if (cmd.Contains("/getusers"))
            {
                int count_users = Server.Clients.Count();
                for (int i = 0; i < count_users; i++)
                {
                    Console.WriteLine("{0}: {1}", i, Server.Clients[i].UserName);
                }
            }
        }


        private static void startServer() // ачало работы сервака; адресовка;
```

```
        {
            IPHostEntry ip_host = Dns.GetHostEntry(localhost);
            IPAddress ip_address = ip_host.AddressList[0];
            IPEndPoint ipEndPoint = new IPEndPoint(ip_address, serverPort);
            Socket socket = new Socket(ip_address.AddressFamily, SocketType.Stream,
ProtocolType.Tcp);
            socket.Bind(ipEndPoint);
            socket.Listen(1000);
            Console.WriteLine("Запущен сервер с IP: {0}", ipEndPoint);
            while(true) // list пользователей
            {
                try
                {
                    Socket user = socket.Accept();
                    Server.NewClient(user);
                }
                catch (Exception e)
                {
                    Console.WriteLine("Ошибка создания клиента: {0}", e.Message);
                }
            }

        }
    }
}
```

//Тест 4-5(пакет chatClient)

```
using System;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using chatClient;

namespace UnitTestProject1
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Mes_1()
        {
            ChatBot at = new ChatBot();
            bool flag = true;
            string ans = "gir";
            string expected = "gir";

            string actual = at.Mes(flag, ans);

            Assert.AreEqual(expected, actual);
        }
        [TestMethod]
        public void Mes_2()
        {
            ChatBot at = new ChatBot();
            bool flag = false;
            string ans = "gir";
            string expected = "введите снова: ";

            string actual = at.Mes(flag, ans);

            Assert.AreEqual(expected, actual);
        }
    }
}
```